Network Watchdog Response Procedures

Network Watchdog Response Procedures

Status: Active runbook
Created: 2026-07-28
Owner: FlowForge / John
Primary design link: `~/system/architecture/network-watchdog-design.md`
Source files verified before writing:

- `~/system/architecture/network-watchdog-design.md`
- `~/system/tools/network-watchdog.sh`
- `~/system/daemons/network-watchdog.js`
- `~/system/daemons/launchagents/com.john.network-watchdog.plist`
- `~/system/docs/incidents/slack-flood-2026-05-15.md`
- `~/system/state/lightrag-ingest-mc/100764.md`

Purpose

Use this runbook when Network Watchdog emits a network, DNS, target-health, or multi-target alert. The goal is to classify the alert, confirm whether it is real or known-noisy, restore the affected target, and escalate before dependent services fail.

Implementation inventory

There are two Network Watchdog implementations in the system tree. Verify which one is active before acting:

1. **Detailed shell checker:** `~/system/tools/network-watchdog.sh`
   - Checks: Internet, FORGE LAN, Tailscale mesh, Azure Vault, BookStack, DNS MX, DNS drift.

- State: `/tmp/network-watchdog-fails/<target>`.
- Log: `~/system/logs/network-watchdog.log`.
- Design reference: `~/system/architecture/network-watchdog-design.md`.

2. **KeepAlive JS daemon:** `~/system/daemons/network-watchdog.js`
   - Checks: default gateway ping, DNS resolution, Internet connectivity.
   - Heartbeat: `~/system/logs/network-watchdog-heartbeat.json`.
   - LaunchAgent file in repo: `~/system/daemons/launchagents/com.john.network-watchdog.plist`.

Important: the design document names `com.alai.network-watchdog`, while current daemon registry files reference `com.john.network-watchdog`. Do not reload or edit LaunchAgents during an incident until the current label/path is verified on the machine.

Alert interpretation

Shell checker levels

Level	Meaning	Default threshold	Response
WARN	First failed check; may self-recover	1 failed run	Confirm with one manual probe, watch next cycle.
ALARM	Sustained failure	Usually 3 consecutive failed runs, about 15 minutes	Start triage and recovery steps.
PANIC	Critical sustained or broad outage	5+ consecutive failures, Internet 2+, or 3+ targets down	Treat as active infrastructure incident. Escalate immediately.

Special cases from verified files:

- DNS MX uses current shell baseline **Migadu** (`migadu.com`), updated after MC #100764. The original design document still mentions `one.com`; use the script as the current executable truth.
- DNS drift Slack alerts are disabled in the shell script because Cloudflare anycast caused false positives.
- Tailscale steady-state offline nodes are suppressed after 100 cycles and reduced to daily heartbeat behavior.

JS daemon levels

`~/system/daemons/network-watchdog.js` alerts after **3 consecutive failures** for gateway, DNS, or Internet checks. It has a 10-minute alert cooldown and sends to Slack plus macOS notification when those channels work.

First triage steps for any alert

1. **Preserve evidence first.** Do not restart services before capturing alert text and recent logs.
2. **Identify implementation and target.** Determine whether the alert came from shell target names (`forge-lan`, `tailscale-mesh`, `azure-vault`, `azure-docs`, `dns-mx`, `dns-drift`, `internet`) or JS target names (`gateway`, `dns`, `internet`).
3. **Read recent logs.**

```
tail -120 ~/system/logs/network-watchdog.log
```

4. **Check current daemon state before touching it.**

```
launchctl list | grep -i network-watchdog || true
launchctl print gui/$(id -u)/com.john.network-watchdog
```

5. **Check counters for shell checker alerts.**

```
ls -la /tmp/network-watchdog-fails
for f in /tmp/network-watchdog-fails/*; do [ -f "$f" ] && printf '%s=' "$(basename "$f")" &&
cat "$f"; done
```

6. **Classify scope.**
   - One target only: likely target-specific.
   - DNS MX or DNS drift only: likely DNS/provider/config issue.
   - Internet + several targets: likely local network or host outage.
   - Tailscale only with high fail count: check whether it is the known steady-state offline-node pattern before paging.

Target-specific recovery procedures

1. Internet connectivity (`internet`)

Alert signals: shell `Internet unreachable`, JS `internet failure`, PANIC after 2+ shell failures.

Confirm:

```
ping -c 2 1.1.1.1
curl -sf --max-time 5 https://1.1.1.1/cdn-cgi/trace
route -n get default
```

Recover:

1. If default route is missing, inspect the active network interface before changing anything.
2. If Wi-Fi/Ethernet is down, restore local connectivity from macOS Network settings or the physical network path.
3. Re-run the confirmation commands.
4. If Internet is down and 3+ watchdog targets also fail, escalate as systemic outage.

2. Gateway (`gateway`, JS daemon)

Alert signals: JS daemon reports `GATEWAY FAILURE` and includes the detected gateway IP.

Confirm:

```
route -n get default
ping -c 3 <gateway-ip-from-alert>
```

Recover:

1. Verify the gateway IP in the alert matches `route -n get default`.
2. If gateway ping fails but Internet works, treat as gateway ICMP filtering/noise and monitor.
3. If gateway and Internet both fail, recover local network path first.
4. Escalate if local network cannot be restored from the host.

3. DNS resolution (`dns`, `dns-mx`, `dns-drift`)

Alert signals: JS `DNS FAILURE`, shell `MX records unexpected`, or shell `DNS drift detected`.

Confirm:

```
dig +short +time=3 google.com
dig +short MX alai.no @1.1.1.1
dig +short A alai.no @1.1.1.1
dig +short A alai.no @8.8.8.8
```

Recover DNS resolution failure:

1. If all DNS queries fail, confirm Internet first.
2. If Internet works but DNS fails, switch/test resolver path before changing app services.
3. Re-run `dig +short +time=3 google.com` and wait one watchdog cycle.

Recover MX failure:

1. Current shell script expects `migadu.com` in `dig +short MX alai.no @1.1.1.1`.

2. If Migadu records are missing, treat as mail-delivery risk.
3. Check Cloudflare DNS for `alai.no` and restore Migadu MX records.
4. Re-run the MX check against 1.1.1.1 and 8.8.8.8.

Handle DNS drift:

- The shell script currently logs DNS drift but disables Slack alerts because Cloudflare anycast caused false positives. If only DNS drift is failing and service endpoints work, record it as noisy unless there is simultaneous service impact.

4. FORGE LAN (`forge-lan`)

Alert signals: shell `FORGE (10.0.0.2) unreachable`.

Confirm:

```
ping -c 3 -W 2000 10.0.0.2
```

Recover:

1. Confirm whether FORGE is expected to be powered on and on the LAN/Thunderbolt path.
2. Check physical link, power state, and host reachability through any secondary access path available at the time.
3. If FORGE is intentionally offline, document the maintenance window and suppress downstream work that depends on it.
4. If FORGE is unexpectedly down for ALARM/PANIC thresholds, escalate to infrastructure owner for hands-on host recovery.

5. Tailscale mesh (`tailscale-mesh`)

Alert signals: shell reports offline node count and names.

Confirm:

```
tailscale status
```

Recover:

1. Identify whether the offline nodes are expected idle/offline devices or required infrastructure hosts.
2. If the alert contains the known historical pattern `makinja-sin-mac-studio`, `basicass-mac-mini`, `iphone181` with a very high fail count, treat as steady-state unless current work depends on those nodes.

3. For required nodes, check Tailscale service on the affected node and re-auth/reconnect only if you have current host access.
4. Re-run `tailscale status` and watch the next watchdog cycle.

6. Azure Vault (`azure-vault`)

Alert signals: shell `Azure Vault unhealthy`, expected HTTP 200/302.

Confirm:

```
curl -s -o /dev/null -w '%{http_code}\n' --max-time 10 https://vault.alai.no/healthz
```

Recover:

1. If HTTP is 200/302, reset/observe; it was transient.
2. If HTTP is 5xx/timeout, check whether Internet and DNS are healthy first.
3. If only Vault is unhealthy, follow the Vault service runbook/host access path and capture HTTP status plus timestamp.
4. Escalate as secrets-access incident if deployments, agents, or BookStack sync are blocked by Vault unavailability.

7. BookStack/docs (`azure-docs` / BookStack)

Alert signals: shell `BookStack unhealthy`, expected HTTP 200/302 for `https://docs.alai.no`.

Confirm:

```
curl -s -o /dev/null -w '%{http_code}\n' --max-time 10 https://docs.alai.no
```

Recover:

1. If HTTP is 200/302, mark transient and watch next cycle.
2. If public docs are down but local BookStack is available, use `~/system/context/docs/runbooks/bookstack.md` for container/API/database recovery.
3. If BookStack API is rate-limited (`429 Too Many Attempts`), stop automation retries and wait for the rate window before sync attempts.
4. If docs are inaccessible during an active incident, preserve this local runbook path: `~/system/docs/runbooks/network-watchdog-response-procedures.md`.

Escalation paths

Escalate based on scope and business impact:

1. **WARN single target:** John/FlowForge watches logs; no CEO interruption unless target blocks current work.
2. **ALARM single target:** John/FlowForge begins recovery. Escalate to hands-on host owner if physical access is needed.
3. **PANIC or 3+ targets down:** Treat as systemic incident. Notify CEO with one factual line: target count, failed target names, first-failure time, and current action.
4. **Mail/DNS MX broken:** Escalate as mail-delivery risk after confirming Migadu MX records are missing from public resolvers.
5. **Secret access blocked:** Escalate if Vault outage blocks deploys, agents, or credential retrieval.
6. **Alert flood:** Do not add more alerts. Apply cooldown/suppression logic first and use the 2026-05-15 incident pattern below.

Historical incident patterns

Verified from design, incident, and MC snapshot files:

1. **2026-04-19 to 2026-04-20 network incident cluster** (`network-watchdog-design.md`): ANVIL OOM with network aspects, `alai.no` MX tampering through Cloudflare, FORGE `10.0.0.2` unreachable from 16:40-17:30, and ANVIL ping disabled before memory fix.
2. **Initial watchdog run 2026-04-20 17:43** (`network-watchdog-design.md`): FORGE was unreachable and Tailscale had 3 offline nodes; Internet, Azure Vault, BookStack, DNS MX, and DNS resolver consistency were healthy at that moment.
3. **2026-05-15 Slack flood** (`slack-flood-2026-05-15.md`): root cause was network-watchdog with 3 permanent-fail checks and zero cooldown. Actions included daemon unloads and network-watchdog disablement decision path.
4. **MC #100764 fix snapshot** (`state/lightrag-ingest-mc/100764.md`): updated MX baseline to Migadu, added per-check 6-hour cooldown files under `/tmp/network-watchdog-lastalert-<check>.ts`, suppressed steady-state Tailscale offline after 100 cycles, and reduced Slack rate from about 60/hour to 1-2/hour.
5. **DNS drift false-positive pattern** (`network-watchdog.sh`): Cloudflare anycast can make 1.1.1.1 and 8.8.8.8 return different A records; the current script logs this and disables Slack alerting for that check.

Post-incident closeout

After recovery:

1. Capture final evidence: alert text, recent `network-watchdog.log` lines, confirmation command outputs, and affected target names.
2. Confirm next watchdog cycle no longer increments the target counter.
3. If a baseline changed intentionally, update both:

- `~/system/tools/network-watchdog.sh`
- this runbook and/or `~/system/architecture/network-watchdog-design.md`

4. If the incident caused user-facing impact, create/update the relevant MC task with evidence and BookStack link.

Related references

- Design: `~/system/architecture/network-watchdog-design.md`
- Shell implementation: `~/system/tools/network-watchdog.sh`
- JS daemon implementation: `~/system/daemons/network-watchdog.js`
- LaunchAgent source: `~/system/daemons/launchagents/com.john.network-watchdog.plist`
- BookStack service runbook: `~/system/context/docs/runbooks/bookstack.md`
- Slack flood incident: `~/system/docs/incidents/slack-flood-2026-05-15.md`
- MC #100764 snapshot: `~/system/state/lightrag-ingest-mc/100764.md`

Revision #2
Created 2026-07-28 14:03:46 UTC by John
Updated 2026-07-28 14:08:19 UTC by John